[資料觀察]

```
####################查看欄位#######################
train.info()
test.info()
```

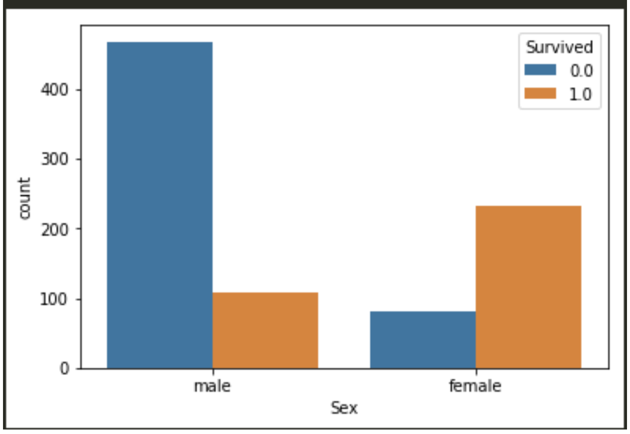
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

查看欄位發現資料中Age, Fare, Cabin, Embarked欄位資料有缺失

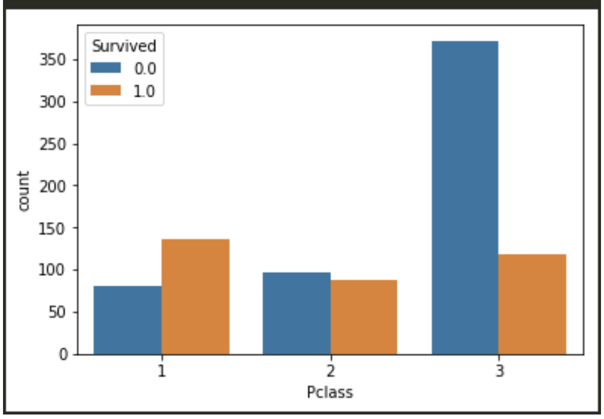
```
############## Observe Data #############
sns.countplot(data['Sex'], hue = data['Survived']) #hue:以色調分類
display(data[['Sex','Survived']].groupby(['Sex'],as_index = False).mean().round(3))
#observation(1):女性存活率大
sns.countplot(data['Pclass'], hue = data['Survived'])
display(data[['Pclass','Survived']].groupby(['Pclass'],as_index = False).mean().round(3))
#observation(2):等級越高，存活率越大
plt.figure(figsize=(50,12))
sns.barplot(y=data['Survived'],x=data['Age'])
#observation(3):小孩或老年存活率較大，尤其小孩更明顯
sns.countplot(data['Ticket'].str[0], hue = data['Survived'])
#obsercation(4):有某些ticket的開頭字母或數字出現率比較高
```

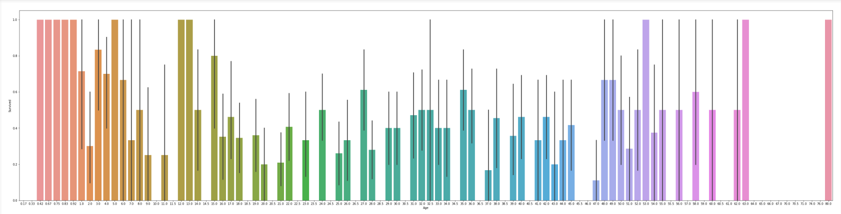
繪出Sex, Pclass, Age, Ticket跟存活的關係圖以利後續處理資料

	Sex	Survived
0	female	0.742
1	male	0.189



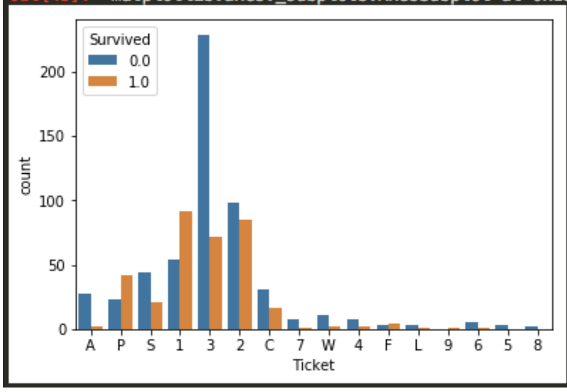
	Pclass	Survived
0	1	0.630
1	2	0.473
2	3	0.242





Age跟Survived關係圖

```
In [45]: sns.countplot(data['Ticket'].str[0], hue = data['Survived'])
Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x1a212611d0>
```



從Ticket跟Survived關係圖可以看出有某幾個字母開頭出現頻率比較高，可能可以把資料簡化成較少種類別

[資料前處理]

```
############## Feature Engineering #############
data['Sex'] = data['Sex'].apply(lambda x: 1 if x == 'male' else 0) #把性別轉成0或1
data['Title1'] = data['Name'].str.split(",",expand=True)[1]
data['Title1'] = data['Title1'].str.split(".",expand=True)[0] #把稱謂擷取出來
data['Title2'] = data['Title1'].replace(regex={'Ms':'Miss',
                                               'Mme':'Mrs',
                                               'Mlle':'Miss',
                                               'Dona':'Mrs',
                                               'Dr':'Mr',
                                               'Major':'Mr',
                                               'Lady':'Mrs',
                                               'the Countess':'Mrs',
                                               'Jonkheer':'Mr',
                                               'Col':'Mr',
                                               'Rev':'Mr',
                                               'Capt':'Mr',
                                               'Sir':'Mr',
                                               'Don':'Mr'}) #把所有稱謂濃縮成四種稱謂
data['Embarked'] = data['Embarked'].fillna('S') #embarked的缺失值用最多人登陸的S港填補
data['Fare']=data['Fare'].fillna(data['Fare'].mean()) #fare用平均數填補
data['Cabin_Letter'] = data['Cabin'].apply(lambda x: str(x)[0])
```

Sex, Name, Cabin欄位做資料簡化 / Embarked, Fare欄位的資料做填補
Name欄位只提取稱謂資料，並簡化成四種主要稱謂，儲存至欄位Title2
Cabin欄位只提取第一個英文字，並儲存至Cabin_Letter欄位

```
for i in [train] :
    i['Age_Null_Flag'] = i['Age'].apply(lambda x: 1 if pd.isnull(x) else 0)
    dt = train.groupby(['Title2'])['Age']
    i['Age'] = dt.transform(lambda x: x.fillna(x.mean())) #age的空值用該稱謂的年齡平均值填補
    i['Fam_Size'] = np.where((i['SibSp']+i['Parch']) == 0 , 'Solo',
                    np.where((i['SibSp']+i['Parch']) <= 3,'Small', 'Big'))
    #把親屬統一成一個欄位，並改以solo,small,big表示
    del i['SibSp']
    del i['Parch']#刪除原本的兩個欄位
```

Age跟稱謂有關係，所以用對應稱謂的年齡平均值填補會比全部平均值填補準確
把SibSp跟Parch兩個跟親屬有關的欄位簡化成一個家庭大小的欄位

```python
i['Ticket_Letter'] = i['Ticket'].apply(lambda x: str(x)[0]) #取ticket的第一個字母或數字做為代表
i['Ticket_Letter'] = i['Ticket_Letter'].apply(lambda x: str(x))
i['Ticket_Letter'] = np.where((i['Ticket_Letter']).isin(['1', '2', '3', 'S', 'P', 'C', 'A']), i['Ticket_Letter'],
                              np.where((i['Ticket_Letter']).isin(['W', '4', '7', '6', 'L', '5','8','9']),'Low_ticket', 'Other_ticket'))
    #根據觀察結果，把較不常出現的字母或數字以'low__ticket'表示
```

把Ticket資料簡化

```python
train['Embarked'] = train['Embarked'].astype('category').cat.codes
train['Pclass'] = train['Pclass'].astype('category').cat.codes
train['Title2'] = train['Title2'].astype('category').cat.codes
train['Fam_Size'] = train['Fam_Size'].astype('category').cat.codes
train['Ticket_Letter'] = train['Ticket_Letter'].astype('category').cat.codes
train['Cabin_Letter'] = train['Cabin_Letter'].astype('category').cat.codes

test['Embarked'] = test['Embarked'].astype('category').cat.codes
test['Pclass'] = test['Pclass'].astype('category').cat.codes
test['Title2'] = test['Title2'].astype('category').cat.codes
test['Fam_Size'] = test['Fam_Size'].astype('category').cat.codes
test['Ticket_Letter'] = test['Ticket_Letter'].astype('category').cat.codes
test['Cabin_Letter'] = test['Cabin_Letter'].astype('category').cat.codes
```

類別資料轉數值資料

[預測]

```python
predictors = ['Age', 'Embarked', 'Fare', 'Pclass', 'Sex', 'Fam_Size', 'Title2','Ticket_Letter','Cabin_Letter']

rf = RandomForestClassifier(criterion='gini',
                             n_estimators=1000,
                             min_samples_split=12,
                             min_samples_leaf=1,
                             oob_score=True,
                             random_state=1,
                             n_jobs=-1)

rf.fit(train[predictors], train["Survived"])
print("%.4f" % rf.oob_score_) #交叉驗證

pred = rf.predict(test[predictors])
```

使用Age, Embarked, Fare, Pclass, Sex, Fam_Size, Title2, Ticket_Letter, Cabin_Letter作為特徵以隨機森林分類器預測存活

[儲存資料]

```python
#################### Save Result ########################
submission = pd.DataFrame({
                             "PassengerId": test["PassengerId"],
                             "Survived": pred
                          })
submission.to_csv('submission.csv', index=False)
```

[上傳結果]